The PI2 Packet Interface User's Guide

Packet Working Group
Ottawa Amateur Radio Club

Revised 14 April 1995

The PI2 Packet Interface User's Guide

Ottawa Amateur Radio Club
Packet Working Group
P.O. Box 8873
Ottawa, ON K1G 3J2
Canada

Limited Warranty

The **PI2** board is warranted to be free of defects in materials and workmanship for a period of thirty (30) days from the date of purchase. In the event of notification within the warranty period of defects in materials or workmanship, the seller will, upon return of the product, repair or replace (at its option) the defective parts. The remedy for breach of this warranty shall be limited to repair or replacement and shall not encompass any other damages, including but not limited to loss of profits, special, incidental, consequential or other similar claims. This warranty does not cover any damages due to accident, misuse, abuse or negligence on the part of the purchaser. This warranty does not cover other equipment or components that a customer uses in conjunction with this product.

THE SELLER SPECIFICALLY DISCLAIMS ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages, or the exclusion of implied warranties, so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

Table of Contents

1. PI2 Card Introduction

The Packet Interface 2 (**PI2**) card is an enhanced version of the original Ottawa PI card, which was introduced in 1990. It is a low-cost high-speed synchronous dual-port serial interface card for packet radio, for use with IBM PC compatibles. It was designed with the high-speed end user in mind, but it can also be used for PC-based packet switches. It provides DMA support for the high-speed port, and interrupt-driven support for a second low-to-medium speed port. DMA support for the high-speed port allows the host system to run additional interfaces, such as the built-in low-speed port, async ports, or ethernet interfaces via the packet driver interface. An onboard timer chip is used to further decrease the interrupt load, and to improve the timing resolution for better efficiency. The PI2 card was designed specifically to run with **KA9Q NOS** and to deliver the full potential of the **GRAPES (WA4DSY)** 56 kbps modem. It also can be used with various 9600 bps and 19.2 kbps modems which are currently available, and it can be used at substantially higher bit rates than 56 kbps, if suitable modems become available. A 1200 bps modem for the interrupt-driven port can be installed on the card. This documentation applies to Revision B of the PI2 card, which began shipping in April 1995.

This document assumes that you have knowledge of the installation and use of the NOS program.

2. Hardware Installation

The first step in installing your PI2 card is to check all of the jumper settings on the card. Some of the settings may have to be changed from the default positions in order to match your external interfacing requirements, or to avoid conflicts with other cards in your system. Then study the interface connection information and prepare the cabling required to connect to your external modem or radio equipment.

Jumper Settings

See Figure 1 for a diagram of jumper placement.

J1 - DCD LED

This is not a jumper, but a two-pin header for connection of a DCD LED indicator if the on-board modem is installed. You may have a spare LED on the front of your PC which can be used for this purpose. The cathode of the LED connects to the pin farthest from the bracket end of the board (i.e., the left-hand side when viewing the board from the component side). There is no harm in connecting the LED backwards - it just won't work!

J2 - TTLRS

This jumper determines whether the A port is configured for TTL or RS422 interface. The standard configuration is for TTL, for which the jumper is in place. To set up the port for RS422, remove the jumper, install a 26LS31 line driver at U14, a 26LS32 or 26LS33 line receiver at U17, and 120 ohm termination resistors at R41-R44.

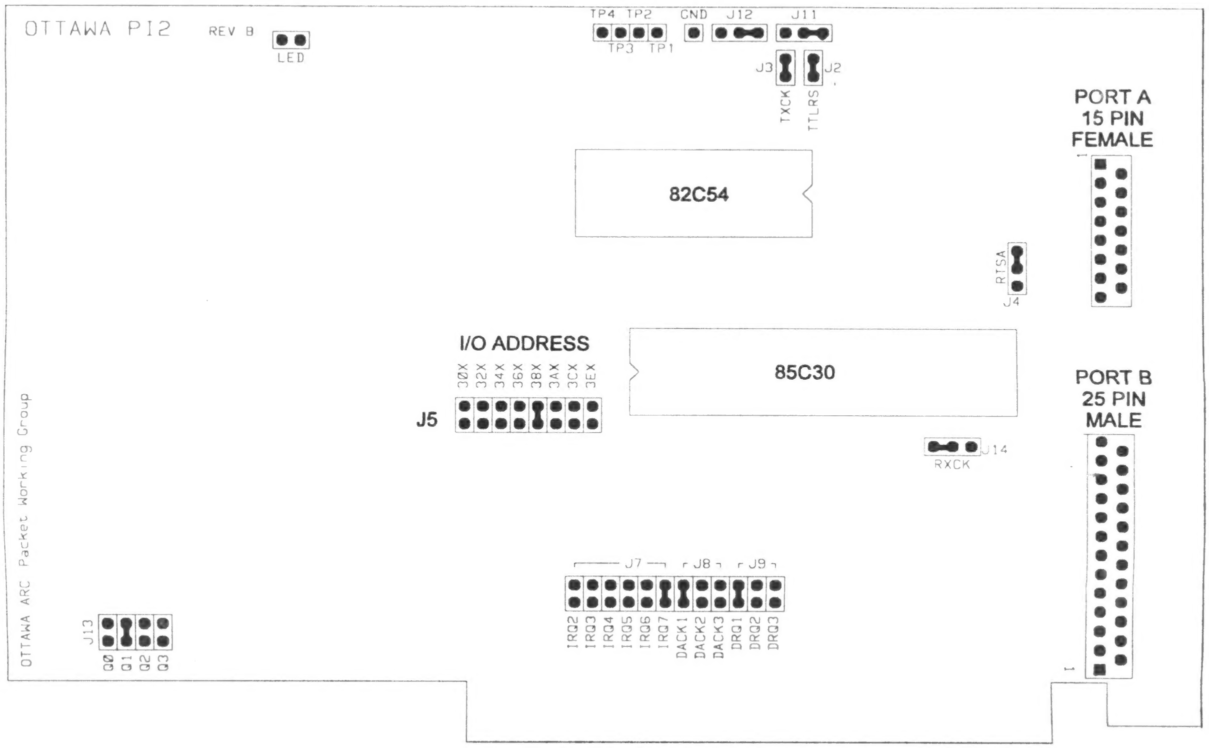


Figure 1 PI2(Rev B) Jumper Locations

J3 - TXCK (Port A)

This jumper determines whether the transmit clock signal from the external modem on the A port is connected to the transmit clock line of the SCC chip. The normal configuration (for the GRAPES modem) is with the jumper in place, and the transmit clock is provided by the modem via the TXC input (P1-4). For modems which require an external transmit clock, this jumper should be removed and the clock signal obtained from the 32CK (P1-8) output on the A port connector. The software driver must also be configured to provide a transmit clock output. One possible use for this facility is to provide the 16X or 32X bitrate clock required for G3RUH-type modems.

J4 - RTSAA

This jumper determines whether the RTS line for the A port will have an open-collector or TTL output. The default is open collector. This is the correct setting for connection to the GRAPES modem - if you use TTL, the modem may go into continuous transmit mode if you power down your PC! The open-collector setting is selected by placing the jumper on the two pins of J4 nearest the top of the board. If the A port is configured for RS422 operation, this jumper should be moved to the TTL position (lower two pins).

J5 - I/O Address Select

The I/O address decoding is set by this jumper block. Only one jumper should be in place in this block. See the section on the memory map (below) for an explanation. The default position is 38X (address 380h), which is the fifth pair from the left on J5.

Note: there is no J6 on the 'Rev B' board.

J7 - IRQ Select

This jumper allows selection of the IRQ signal to be used by the card. The numbers 2 through 7 correspond to the IRQ line desired. The following may help in the selection of the IRQ to be used for the PI2 card:

IRQ2: Likely to be available. Note that in AT bus (286/386/486) systems, you should use '9' rather than '2' in the attach command for the PI2 in NOS, due to the way in which the interrupt controllers are cascaded.

IRQ3: Likely to be available if the COM2 async port is not used.

IRQ4: Likely to be available if the COM1 async port is not used.

IRQ5: Likely to be available. It may be used by the hard drive in some XT systems, or by LPT2 if you run OS/2.

IRQ6: Not generally available (used by the floppy controller).

IRQ7: [Default] Likely to be available. It is used for LPT1 under OS/2, but not under DOS. However, most parallel port boards have a connection for the IRQ, and you may have to disable this connection to prevent interference with the operation of the PI2 (same applies to IRQ5 if you have two parallel ports).

If you're having difficulty identifying a free IRQ for the PI2 board, keep in mind that many boards (e.g., serial port boards) are now available which allow use of the IRQ's above 7 on the AT bus.

J8, J9 - DACK and DRQ

DACK and DRQ are used to select the desired DMA channel. The default is channel 1, which is normally available, unless you have another card in your system which uses DMA. Another possible choice is channel 3. Channel 2 is usually used for floppy disk I/O. Both jumpers must be set to the same channel number.

Note: there is no J10 on the 'Rev B' board. The following jumpers were added in Rev B:

J11 - Port B 16X/32X Transmit Clock Select

The PI2 is wired to provide a 16X bitrate clock output on the B port transmit clock line (P2-24). If you require a 32X clock instead, a 1X3 pin header can be installed in this position to allow selection of either clock rate by means of a jumper (or just install a wire jumper , if you think you'll never need anything other than the 32X clock). A track must be cut (on the component side of the board) in order to allow changing from the 16X rate. If the header is installed, place the jumper on the righthand two pins (viewed from the component side of the board) to get the 16X clock, and on the lefthand two pins to get the 32X clock. Please note: if your external

modem requires a transmit clock to be supplied *at the bitrate* rather than at 16X or 32X, the jumper at J11 should be installed in the 32X position.

J12 - Port A 16X/32X Transmit Clock Select

This is set up in exactly the same way as for the B port (see above): the board comes with 16X clock as the hardwired default. Note that if you want the transmit clock to be an output rather than an input, you must also remove the TXCK jumper (J3). Please note: if your external modem requires a transmit clock to be supplied *at the bitrate* rather than at 16X or 32X, the jumper at J12 should be installed in the 32X position.

J13 - DDIOW Delay Select

This header provides a means of changing the delay of the DDIOW signal, to accommodate motherboards with unusual bus timing. The default position of the jumper is Q1 (second from the left). The most common symptom of bus timing problems is a failure to transmit decodable packets, while receiving packets is okay. If this happens, try moving the jumper to the next position (Q2), and if that doesn't help, try the Q3 position.

J14 - RXCK (Port A Receive Clock Select)

The PI2 is wired to accept a receive clock signal from an external modem modem on the A port. The standard driver software also permits selecting internal clock recovery, using the DPLL in the SCC chip. However, the maximum rate for operation in this mode is 57.6 kbps. This jumper position allows the A port receive clock input to be connected directly to the 7.3728 Mhz onboard clock, which then permits internal clock recovery at 230.4 kbps. To activate this option, a track must be cut on the solder side of the board and a jumper installed in the opposite position.

External Connections - Port A

The A port has DMA support for high speed operation. It is the DA15 female connector. The standard port configuration is for TTL-level interface. The port can be reconfigured for RS422 interface (see Appendix D), which is highly recommended for high-speed operation, especially when the modem is more than a few feet away from the PI2 card.

The A port (P1) pinout is as follows:

Pin 1 - **PTT Push-to-talk output.** This is an open-collector "push-to-talk" output. It can withstand about 30 volts. It is intended to be used to key a transverter, but it has no current-limiting. The associated transverter must therefore limit current to about 150 milliamps.

Pin 2 - **RXC Receive clock input**. This is a TTL-level (or RS422, if that option is enabled) input from the modem.

Pin 3 - **RXD Receive data input.** This is a TTL (or RS422) input from the modem.

Pin 4 - **TXC Transmit clock input.** This is a TTL (or RS422) input when the port is configured for external clocking (see the **attach** command).

Pin 5 - **$\overline{\text{DCD}}$ Data carrier detect input.** This is a TTL (or RS422) input that receives carrier detect from the modem. Note that this input is active low. If the RS422 option is used, the signal will be inverted.

Pin 6 - **TXD Transmit data output.** This is a TTL (or RS422) output that provides transmit data to the modem.

Pin 7 - **$\overline{\text{RTS}}$ Request to send output.** This line can be configured to provide TTL (or RS422) level, or open-collector output. It is usually set to open-collector output when driving a **GRAPES** modem, because this prevents a continuous transmit condition when your computer is turned off. Note that this output is active low when configured for TTL output, and if the RS422 option is used, the signal will be inverted.

Pin 8 - **32CK Transmit clock output.** This is a TTL-only line which provides a clock for modems which require an external clock source. The clock rate is determined by the driver software, but the usual use for this output is to provide a 16X or 32X bitrate clock for the state machines in G3RUH-type modems (the name 32CK is a slight misnomer, since the default rate is now 16X).

Pin 9 - **Ground.**

Pin 10 - **RXC(-) Receive clock input.** Paired with pin 2, this is the other side of the balanced input when the port is configured for RS422 operation.

Pin 11 - **RXD(-) Receive data input.** Paired with pin 3, this is the other side of the balanced input when the port is configured for RS422 operation.

Pin 12 - **TXC(-) Transmit clock input.** Paired with pin 4, this is the other side of the balanced input when the port is configured for RS422 operation.

Pin 13 - **DCD(-) Data carrier detect input.** Paired with pin 5, this is the other side of the balanced input when the port is configured for RS422 operation.

Pin 14 - **TXD(-) Transmit data output.** Paired with pin 6, this is the other side of the balanced output when the port is configured for RS422 operation.

Pin 15 - **RTS(-) Request to send output.** Paired with pin 7, this is the other side of the balanced output when the port is configured for RS422 operation, and the RTS jumper is not set for open-collector output.

Please note: The older **PI** card had pins 9 through 14 grounded. The PI2 card has only pin 9 grounded, and 10 through 14 are reserved for RS422 connections. If you are using the standard TTL interface and want your modem cable to be compatible with both the PI and PI2 A port connectors, make sure that you use only pin 9 for ground connections.

External Connections - Port B

The connector for the B port is the DB25 male. It can be used either to connect an external modem using TTL signals, or to connect a radio to the on-board 1200 bps modem if it is installed. If an external modem is used, a 74LS244 should be installed at U20, and U10 (TCM3105) and U16 (74HC132) should not be installed.

Pin 1 - **Ground.**

Pin 2 - **TXD Transmit Data output .** This is a TTL output which provides transmit data to the external modem.

Pin 3 - **RXD Receive Data input.** This is a TTL input for received data from the external modem.

Pin 4 - **$\overline{\text{RTS}}$ Request to Send output.** This is a TTL output which tells the external modem when to enter transmit mode. Note that this output is active low.

Pin 7 - **Ground.**

Pin 8 - **$\overline{\text{DCD}}$ Data Carrier Detect input.** This is a TTL input from the external modem which indicates the presence of valid received data. This input is active low.

Pin 11 - **RXA Receive audio input.** This is a single-ended low-impedance input that is normally connected to the speaker output of a radio. This signal can be observed at test point TP1. The nominal input level at this point is about 0.3 volts (p-p); it should not exceed 0.75 volts p-p.

Pin 12 - **PTT Push-to-talk output.** This is an open-collector output for keying transmitters. It can withstand about 30 Volts, and can sink about 150 mA.

Pin 13 - **TXA Transmit audio output.** This is a single-ended high-impedance output that is normally connected to the microphone input of a transmitter. The level is dependent on the setting of R26, the TX AUDIO adjustment. It can provide a maximum of about 1.5 Vp-p into a high-impedance load. If the resistor R28 is installed, this line can also serve to key the transmitter in radios (usually handhelds) which combine both functions on one line.

Pin 15 - **RXC Receive Clock input.** This is a TTL input for the recovered clock from an external modem, when external clocking mode is used.

Pin 24 - **TXC Transmit Clock output.** This is a TTL output for supplying a 16X or 32X bitrate transmit clock to an external modem.

More details on installing and using the on-board modem can be found in Appendix B.

I/O Address and Memory Map

The base address of the card may be set to 300h, 320h, 340h, 360h, 380h, 3A0h, 3C0h, or 3E0h. These correspond to positions 30X through 3EX on the jumper block. We recommend using the default address of 380h. Some of the other addresses are likely to result in conflicts. For example, 3C0h is used by EGA/VGA video cards, and the upper part of the 3E0h block is used by the COM1 serial port. The 300h address is very commonly used as the default address in ethernet adapters and other types of cards.

The offsets of the registers from the base address are:

0 - B port control register
1 - B port data register
2 - A port control register
3 - A port data register
4 - DREQ mask register. bit0=DREQ mask control
8 - Timer 0 register
9 - Timer 1 register
A - Timer 2 register
B - Timer command register

Note: this information is not needed for operation of the card; it is simply provided for the benefit of those who may be curious about some of the technical details.

3. Software Configuration

Using the Internal PI Driver for KA9Q NOS

The *attach* Command

The NOS program must be made aware of the card through the use of the attach command. The syntax of this command is:

attach pi *<address> <vector> <dmachannel> <mode> <label[$label]> <buffers> <mtu>*
<speed a[g]> <speed b[g]> [ip_addra] [ip_addrb]

The parameters in angle brackets are required, and those in square brackets are optional. The parameters for *address*, *dmachannel* and *vector* must correspond to the jumper settings on the card.

The *mode* must be set to **ax25** as that is the only mode the PI2 card currently supports.

The *label* parameter provides names for the two PI2 interfaces. It does this in either of two ways, depending on whether this parameter contains a "$" character. If there is no "$", then the name will be automatically modified by appending an "a" or a "b" to distinguish the high speed (A) port from the low speed (B) port. For example, the label "pi0" would result in two interfaces, one called "pi0a", the other called "pi0b". If the parameter contains a "$", the portion before the "$" becomes the name of the A port, and the portion after it becomes the name of the B port. For example, setting the parameter to "56k$1200" would result in the two interfaces being named "56k" and "1200", respectively. [Note: this independent-naming feature is only present in PI software drivers dating from November 1993 or later].

The *buffers* parameter specifies the size, in bytes, of the receive buffer used by the driver. It should be equal to the *mtu* value (see discussion below).

The *mtu* parameter specifies the largest packet, in bytes, that the driver should transmit. It is very important that this be at least 40 bytes larger than the largest TCP MSS that you plan to use (see discussion below).

The *speed a[g]* parameter is the speed of the A port in bits per second. If you specify 0 for this parameter, the card will be configured for external clocking on the A port. The maximum speed which can be selected for internal clocking is 57,600 bps (Note: this can be increased to 230.4 kbps. See the note on installing jumper J14 in the section on jumpers. Changes to the driver are also required - contact us for details). In internal clocking mode, you can also attach a "g" flag to this parameter (e.g., "9600g"). This puts the port in "G3RUH mode". In this mode, the card provides a transmit clock at 16X the bit rate (or 32X - see the note on jumper J12 in the section on jumpers) on the 32CK output (P1 pin 8). If the flag is not present, the transmit clock will be at the bit rate (if J12 is in the 32X position). **Please note** that if you use internal clocking, you should remove the TXCK jumper (J3), so that the TXC input buffer (part of U15B) will not be driving the TXCK line.

The *speed b[g]* parameter is the speed of the B port in bits per second. If you specify 0 for this parameter, the card will be configured for external clocking on the B port. Since this port is interrupt-driven, the maximum bit rate which can be achieved without excessive overruns is highly dependent on the speed of the host PC, as well as the interrupt latency of the host software and the load presented by other interrupts on the system. It is not guaranteed to work at rates beyond 1200 bps. The "g" flag has the same function as on the A port, causing a 16X bitrate clock (optionally 32X, if jumper J11 is installed - see the section above on jumpers) to be output on the transmit clock line (P2-24). If you are using the on-board modem, this parameter should be set to "1200".

The *ip_addr* parameters are optional. If they aren't specified, the IP addresses of the corresponding ports will be set to whatever the system-default IP address is.

Here is an example **attach pi** command for a **GRAPES** modem running at 56 kbps (which provides its own clocking), and a secondary port with a 1200 bps modem with clock supplied by the PI2 card. Both interfaces will use the default IP address, and they will be named **pi0a** and **pi0b**.

attach pi 0x380 7 1 ax25 pi0 2088 2088 0 1200

For a G3RUH-type 9600 bps modem on the A port, the attach line might look something like this:

attach pi 0x380 7 1 ax25 9k6$1k2 552 552 9600g 1200

In this example, we also show the use of the alternate form of the *label* parameter, to name the two PI2 interfaces **9k6** and **1k2**.

As mentioned above, the maximum bit rate that can be used with internal clocking is 57,600 bps (with the exception of the modification for 230.4 kbps, as noted above). This limitation is imposed by the clock requirements of the SCC DPLL (32X bit rate) which is used for receive clock recovery in this mode. The speeds that can be selected are given by:

speed = (115,200)/(x + 2)

where x = 0, 1, 2,...

Allocation of Buffers for the PI2 Interface

The first step in setting the parameters related to memory usage by the PI2 interface is to decide what TCP MSS you will use. In general, larger values are more efficient because they generate longer transmissions and therefore less overhead. On the other hand, there will be longer delays when retransmissions are required, so the optimum MSS depends on the reliability of the links. Long transmissions may also become less desirable when several users are sharing a channel, especially at low bit rates. Therefore it is hard to generalize, since every situation is a little different; the optimum has to be determined experimentally. At 56 kbps, a TCP MSS of 2048 is a good starting point; for 19.2 kbps and 9600 bps, you might want to start with 1024 and 512, respectively. The MSS is the maximum amount of <u>data</u> that can be in one TCP segment; it does not include the headers and other overhead added by the protocols at the layers under TCP.

The MTU (Maximum Transmission Unit) of an interface sets the maximum size of a packet that can pass through that interface. It includes the headers which are added by the TCP and IP layers, but not the overhead from the link layer (e.g., AX.25). The TCP and IP headers contribute 20 bytes apiece, so the MTU must be at least 40 bytes larger than the maximum TCP MSS you plan to use. Thus if you use a TCP MSS of 2048, you should set the MTU parameter in the PI2 attach to 2088.

The buffer whose size is specified in the PI2 attach must be large enough to hold a packet equal to the MTU size. Therefore this parameter is normally set to the MTU value. The only reason this parameter exists at all is to support some versions of NOS (e.g., vanilla KA9Q) that do not have a preallocated pool of interrupt buffers. So, set this to the MTU value and forget it.

If you find that your PI2 attach fails, complaining about insufficient memory, it may be because there is a *mem ibufsiz* statement in your autoexec.nos file which sets the interrupt buffer size to something smaller than you require for the PI2 interface. This sets the size of the buffers in the interrupt buffer pool mentioned above. The PI driver requires a small amount of additional overhead (currently 6 bytes) when it uses these buffers, so the buffers must be slightly larger than the *buffers* parameter you specify in your PI2 attach statement. If this is not the problem, then it is likely that the interrupt buffer size specified at compile time was too small. You will either need to change the define (IBUFSIZE in config.h) and recompile, or reduce the buffer size and MTU values in your attach (and reduce TCP MSS too, of course). The default *mem ibufsiz* in the PI2 software distribution is 2100, which will work fine with the MTU of 2088 (maximum TCP MSS of 2048) given in the example in the previous section. If you want to try a larger MTU, you should recompile with a larger IBUFSIZE rather than attempt to increase it with a *mem ibufsiz* statement in your startup file (it is safe, however, to use this technique to make the *ibufsiz* smaller than the default value).

One final parameter to be aware of is the <u>number</u> of interrupt buffers in the buffer pool. It is also set at compile time by a define in config.h, but you can select a different value (smaller or larger) using the *mem nibufs* command in your autoexec.nos file. The default value in the PI2 distribution software is 10, but you can try experimenting with smaller values (and thus saving a bit of memory). After running for awhile, check the number of *Ibuffails* by means of the *mem status* command; if there are more than a few (and the number increases with time), you should allocate more buffers to the pool.

The *param* Command

This command allows you to set various operational parameters. It is available only with the internal PI driver for NOS, not the external packet driver (in the latter case, the parameters are set when the packet driver is loaded, and cannot be changed during operation). The syntax of this command is:

param ***interface parameter-number value***

Using the previous example, entering:

param pi0a 1 15

would set parameter 1 for the interface name pi0a to a value of 15. The usage is similar to that used for asynchronous KISS interfaces in NOS (but note the difference in units for the A port).

In the following, all time parameters are in units of 1 millisecond for the A port, and units of 10 milliseconds for the B port:

param 1 - TXDelay

This sets the time from transmitter keyup until data is sent. HDLC flags are sent during this time. The default for the A port is 15 (15 ms) and the default for the B port is 30 (300 ms).

param 2 - P-persistence

This sets the probability that a transmission will be attempted when the channel is free and data is available. Setting it to 64, for example, will give a 64/255 probability. Setting it to 255 will defeat P-persistence. The default setting for this parameter is 128, giving a 50% probability.

param 3 - Slot Time

This determines the amount of time that will pass before a transmission which has been deferred due to P-persistence will be attempted again. The default setting for the A port is 15 ms, and for the B port it is 100 ms.

param 4 - Tail Time

This determines the amount of time the transmitter will remain keyed after a packet has been sent. Because this time is measured from the time that the CRC has been sent, it must be long enough to allow the closing flag to be sent. In previous releases of the PI driver, this parameter was set to defaults of 1 (1 ms) for the A port and for 3 (30 ms) for the B port. The 1 ms default for the A port was fine for operation at 56 kbps, but for 9600 bps or 19.2 kbps, it needed to be increased to 3 for reliable operation. Beginning with the January 1994 release, the driver now sets this parameter automatically, depending on the speed of the associated interface. The 'param 4' command, if used, will still override the automatically set value.

The *pistatus* Command

The syntax of the **pistatus** command is:

pistatus [show|clear]

The *pistatus show* command displays a summary of statistics about the PI2 card. These include (in order of presentation):

The number of receive, transmit, and external interrupts, frames enqueued for transmission, frames received correctly, frames received with CRC errors, receive FIFO overruns, number of transmit underuns. The statistics are shown for both ports of the card.

There are some apparent anomalies in the status display which are actually normal and do not mean that there is a problem with the card or the software:

The transmit interrupt count will remain at zero for the A port, because DMA operation does not make use of transmit interrupts. It is quite normal for the *Crcerr* and *RxOvrs* counts to accumulate. These conditions are typically generated by the noise bursts received at the end of a transmission. Transmit underuns should be a fairly rare occurrence, however. Finally, be aware that after the card has been running for a while, some of the statistics may wrap around to zero, and show negative numbers.

The *pistatus* command was mainly intended for debugging use by developers; however, it can sometimes provide useful clues for troubleshooting installation problems. For example, if you have a problem with transmitting on a PI2 interface, check the *Tstate* for that interface. If it seems to be stuck in the DEFER state, it means that the driver has data queued for transmission, but it thinks the channel is busy. This may indicate that the modem is providing a continuous DCD indication, or that there is a problem with the interface wiring.

The *pistatus clear* command resets all of the counters to zero.

Note that the *pistatus* command is only available with the internal PI driver for NOS, not with the external packet driver.

Using the PI Packet Driver

Included in the software support package for the PI2 board is a packet driver which is compliant with the specification from FTP Software, Inc. It has been tested with several NOS variants and found to be functional. It should be noted that this is an AX.25 class driver, so any application used with this driver must be able to understand AX.25 formatted packets. In other words, the only practical use for it right now is to enable you to use variants of NOS which do not have PI card support compiled-in.

There is an exception to this rule. The driver may be re-assembled to provide "simulated ethernet" support, i.e., the card can be made to look like an ethernet card to application software. We have no plans to support or further enhance this particular mode of operation, but if you find it of interest, see the comments in pi.asm for details.

Limitations of this release of the PI packet driver:

1. Only the A port (high speed port) is supported.

2. Unlike the built-in driver, parameters must be specified on the command line at the time the driver TSR program is installed. They can't be changed while NOS is running.

3. There is no *pistatus* command.

Install the driver (**pi.com**) with the command:

pi *<packet_int> [hw_int] [io_addr] [dma] [baud] [TXD] [P] [slot] [tail] [clkmode] [bufsize] [buffers]*

The parameters are as follows:

packet_int: Software interrupt number the driver uses to communicate with the application. It should be in the range from 0x60 to 0x7F.

hw_int: The hardware interrupt in use. (set by the IRQ jumper J7 on the board).

io_addr: The address of the PI2 card I/O registers (set by jumper J5).

dma: The DMA channel in use (set by jumpers J8 and J9). Usually channel 1.

baud: The desired baud rate when internal clocking is used. Set to zero if external clocking is desired (as when using the GRAPES modem).

TXD: The transmit delay time in milliseconds. This is the time during which flags are sent before the actual data is transmitted.

P: Probability factor for the P-persistance algorithm (0-255).

slot: Slot time in milliseconds for the P-persistance algorithm.

tail: Delay time to allow the CRC and closing flag to be transmitted before the transmitter is turned off. This value is calculated by the driver if internal clocking is used, but may be over-ridden from the command line if desired (value in milliseconds).

clkmode If internal clocking is used (the baud parameter is not 0), this parameter gives further control over the clocking. If it is 0 (default), the receive clock is recovered from the received data, and NRZI coding is used. This mode is suitable for driving a modem which does not provide external clocking signals. The maximum baud rate in this mode is 57,600 bps.

If clocking mode is set to 1, a clock signal is output on A-TXC, while receive clocking is derived from A-RXC. NRZ coding is used. This mode is useful for connecting PI2 cards back-to-back for example, because higher baud rates can be obtained. A secondary effect of choosing mode 1 is that the baud rate is 32 times that specified by the baud parameter.

bufsize: This parameter allows you to specify the amount of memory to reserve for DMA buffers. The default size is 2048.

buffers: The number of DMA buffers allocated to the driver. The default is 5, and the minimum is 3.

For example:

pi 0x7e 7 0x380 1 0 15 128 10 1 0 2048 5

If any parameters after the first one are left off, defaults as shown in the above example will be used. These are typical parameters for use with the **GRAPES** modem. If the first one is left off, the driver will complain.

In NOS, the attach command in your autoexec.nos file should look like this:

attach packet *<packet_int> <label> <buffers> <mtu> [ip_addr]*

The parameters are:

packet_int: The software interrupt specified when the packet driver was loaded.

label: The name which you want to attach to your PI2 A Port interface.

buffers: The maximum number of packets which will be allowed on the transmit queue.

mtu: The maximum size of packet, in bytes, that the driver should transmit.

ip_addr: Optional IP address to associate with this interface. If unspecified, the default address will be used.

For example:

attach packet 0x7e pi0a 1 2000 44.1.2.3

Make sure that you use a TCP MSS which is at least 40 bytes less than the MTU. Also, best results will usually be obtained with TCP WINDOW = TCP MSS on half-duplex channels, but you can experiment with the window size set to larger multiples of MSS to verify this.

4. Error Messages

The PI2 card driver will generate error messages under certain failure conditions. They are:

PI: ERROR - DMA page boundary violation

This is an internal error.

PI: Mode *string* **unknown for interface** *string*

The attach command will generate this error if any mode other than **ax25** is specified.

PI: Interface *string* **already exists**

The interface name chosen is not unique.

PI: Set mycall first

Generated by the attach command if **ax25 mycall** has not yet been set.

PI: IRQ # out of range

An invalid IRQ number has been used in the attach command.

PI: No IP address

No IP address has been supplied to the attach command for one or both ports, and there is no default.

PI: No memory available for receive buffers, or <buffers> parameter too large

Memory is so low that the attach command has failed, or you specified a buffer size greater than the maximum allowable (6 bytes less than the previously defined *ibufsize*).

PI: No memory available for transmit buffer

Memory is so low that the attach command has failed, or you specified too large an MTU.

PI: Insufficient parameters

The param command has been supplied an insufficient number of parameters.

PI: Parameter string out of range

The param command has been issued with an invalid parameter.

PI: Maximum of 3 PI cards supported

There are not enough 8-bit DMA channels to support more than 3 cards.

5. Bug Reports and Software Updates

If you think you have found a bug, please document it with a full description of the circumstances and report it to:

bm@hydra.carleton.ca

Or to: Packet Working Group
Ottawa Amateur Radio Club
P.O. Box 8873
Ottawa, ON, Canada KlG 3J2

Include a description of your hardware setup, software version, and the conditions or actions which led up to your problem. We would also appreciate reports of errors in this documentation, or suggestions for improvements, preferably sent to the above e-mail address.

The latest PI and JNOS software is available via anonymous FTP from hydra.carleton.ca. Check the directory pub/hamradio/packet/tcpip/pi2 and other nearby directories. If you don't have Internet access, find someone who does! As a last resort, you can contact us at the above address, and we'll let you know if there are any updates available, and make arrangements to get them to you.

And, lastly, if you are interested in developing a PI2 driver for another OS, we would like to hear from you!

Appendix A - Hardware description

The PI2 card uses a 85C30 dual port Serial Communications Controller (SCC). This chip has been described as the Swiss army knife of serial chips. It does require a bit of glue logic to interface it to the ISA bus, however. U1 is a bidirectional bus buffer used to meet the loading and drive requirements of the bus. U4 and U11-F provide buffering of additional bus signals. U9 and U11-A and -B are used to delay the leading edge of the buffered I/O write signal for the SCC. U3, a 74LS138, and the 16L8 PAL, U2, provide all address and chip select decoding. U6-A and U5-A provide a means of masking DMA requests independent of the internal registers of the 85C30.

The 82C54 timer, U8, provides an independent timer for each serial port. One timer in this chip is used as the prescaler for the remaining two. The timer outputs are connected to the CTS pin of each serial port. In this way, the CTS interrupts can be used to time events such as TXDELAY. The SCC receives its clock directly from the 7.3728 oscillator, while the timer is clocked at half this frequency via U6-B.

On the A port interface, all of the TTL signals are buffered by U15. The RTS output can also be selected to be open-collector (Q4), and there is a separate open-collector PTT output (Q3) available. Both outputs are controlled by a watchdog timer which will terminate transmission if the RTS signal is asserted continuously for more than about 10 seconds. The port can also be configured for RS422 operation if U14 and U17 are installed. The receive data (RXD) signal is qualified by the DCD signal in U5-B, which prevents any interrupt load to the PC's processor during no-signal conditions. The RXD signal is also qualified by RTS, which prevents transitions on the SCC RXD input during transmit mode. This keeps the SCC DPLL from being perturbed so that it can be used as the source of a stable external transmit clock, for modems which require this.

The low-speed B port has optional TTL buffering for external modems, if U20 is installed. This chip should not be installed if the on-board 1200 bps modem is installed. The modem is a standard TCM3105 design, similar to that used in many other packet interfaces. Unlike many of the other interfaces, however, it also includes an external DCD circuit for effective "open squelch" operation without excessive falsing. This provides faster carrier sensing, leading to fewer collisions and more efficient CSMA (Carrier Sense Multiple Access) operation. The DCD circuit is based on the TAPR design, and the EPROM U13 contains a modified version of the TAPR STATE109 code. As in the A port, this port provides an open-collector PTT output which is regulated by a watchdog timer, and the RXD line is qualified by DCD and RTS.

Appendix B - Installation and Use of the On-Board Modem

Parts List

Begin by checking the contents of your parts kit. It should contain the following:

Qty	Description	Designator	Comments
[ ] 1	TCM3105NL	U10	IC, 1200 bps Modem
[ ] 1	74HC132	U16	IC, Quad Schmitt NAND
[ ] 1	74HC374	U12	IC, Octal Latch
[ ] 1	27C64	U13	IC, 8Kx8 Eprom **[**Note 1]**
[ ] 4	1N4148	D3,D4,D5,D6	Diode
[ ] 1	2N3906	Q2	Transistor
[ ] 1	4.433618 MHz	Y2	Crystal

Capacitors (0.1" lead spacing):

[ ] 2	33pF	C21,C22	Ceramic (marked 33 or 330)
[ ] 6	0.1uF	C13,C23, C24,C25, C27,C47	Ceramic (marked 104)
[ ] 1	0.47uF	C26	Ceramic (marked 474)

Resistors (1/4 watt, 5%):

[ ] 2	100 ohms	R24,R25	(brown-black-brown-gold)
[ ] 1	220 ohms	R32	(red-red-brown-gold)
[ ] 1	2.2K ohm	R28	(red-red-red-gold) **[**Note 2]**
[ ] 1	4.7K ohm	R31	**[**Note 3]**
[ ] 1	15K ohm	R23	(brown-green-orange-gold)
[ ] 1	33K ohm	R22	(orange-orange-orange-gold)
[ ] 1	100K ohm	R27	(brown-black-yellow-gold)
[ ] 1	220K ohm	R30	(red-red-yellow-gold)
[ ] 1	470K ohm	R29	(yellow-violet-yellow-gold)
[ ] 2	50K ohm trimpot	R26,R45	Multiturn, side adjust (in-line leads, 0.1" spacing)

Hardware:

[ ] 1	4x1 Header	TP1-4	Test Points
[ ] 1	2x1 Header	LED	For Connection to External LED
[ ] 1	1x1 Header	GND	Ground connection for testing
[ ] 1	14 Pin IC Socket		For use with U16
[ ] 1	16 Pin IC Socket		For use with U10
[ ] 1	20 Pin IC Socket		For use with U12
[ ] 1	28 Pin IC Socket		For use with U13

Note 1: This eprom contains a modified version of the TAPR "STATE109" code, identified as "STATE109PI2" (also included in binary form on the PI2 distribution disk). A standard

STATE109 eprom will not work in this application! The state machine code is used and distributed with the permission of TAPR.

Note 2: Installed only for certain radios (see text below)

Note 3: Usually coded yellow-violet-red-gold, but the resistors supplied with the kit may be 1% values, coded yellow-violet-black-brown-brown, on a blue body.

Modem Assembly

Required Tools:

Low-wattage soldering iron with a fine tip (preferably a temperature-controlled soldering station)
Side cutters
Needlenose pliers

Test Equipment:

Voltmeter (DC and AC)
An oscilloscope is very helpful for doing the final bias adjustment, and for checking audio level and distortion.

The modem parts are not difficult to install, but if you have some doubts about your soldering abilities, get some help! Also, since this kit (and the PI2 card itself) contains CMOS components, you must take precautions against static electricity damage during assembly and subsequent handling of the board. At the very least, you should touch a grounded metal object before handling the board or the components.

Please note: Resistor R28 (2.2K) should *only* be installed if you will be using a radio with the microphone audio and PTT combined on one connector.

Refer to the parts placement diagram at the end of this manual. The modem parts to be installed are shown in the shaded areas of the diagram. Begin by installing the four IC sockets. Make sure that the notches in the sockets correspond to those of the outlines on the board's silkscreen (with the exception of the 82C54 timer chip, all of the ICs on the PI2 card have 'right side up' orientation). Take care to keep the sockets firmly seated against the board while you solder their pins. Most people prefer to "tack solder" the corner pins first, while holding the socket in place.

Now you can install the other components (except the ICs), using the existing components on the board as a guide. All of the resistors and diodes are mounted vertically. Be careful to install the diodes correctly: the cathode (vertical stroke) on the diode symbol on the silkscreen corresponds to the end of the diode marked with a band. The cathodes face towards the bottom of the board. The crystal can be mounted upright or flat against the board; it is less vulnerable to damage in the flat position, but it must be insulated from the tracks on the top of the board (e.g., with a piece of electrical tape). Also, if you mount it flat, do not bend the leads sharply; bend them on a radius, some distance from the body of the crystal. Take care to keep the headers at right angles to the board while you solder them in place. Make sure you don't confuse the 0.1 μF and 33 pF capacitors, as they may look very similar. Use a magnifier if necessary to check the values marked on them.

When you have all of the components installed, check the underside of the board carefully for solder bridges and unsoldered pins, and double-check the location of the components. Lastly, insert the four ICs into their sockets, again observing static handling precautions. Once the modem is checked out, you may want to remove the flux from the solder side of the board, using a suitable solvent such as alcohol.

Modem Adjustments

Transmit Adjustments

The first thing to do is check that your radio is keying properly. Reduce the transmit audio level to minimum by turning the adjustment screw on R26 counterclockwise until it clicks against its stop. With the radio connected to a dummy load or tuned to a quiet frequency, make some test transmissions and see if the radio keys up. There are a number of ways you can make NOS transmit on this port - assuming the interface is called pi0b, you could, for example, attempt an AX.25 connection to a nonexistent station:

c pi0b test

The transmissions will continue until you reach the 'ax25 retry' maximum, or you use 'reset' to close the session. This method produces rather short transmissions. A more convenient method of generating test transmissions is the beacon capability in JNOS. To set it up, enter the following:

ax25 bctext "Text [the more you put here, the longer the transmission]"
ax25 bcport pi0b

Now, every time you enter

ax25 bc pi0b

it should cause a transmission to occur (don't forget that JNOS has a command recall capability: just hit up-arrow to retrieve the previous command for execution again). You can set up a repetitive transmission by entering

ax25 bcinterval <seconds>

You can stop the transmissions by setting the interval to zero.

Yet another way of producing transmissions on a given interface is to ping a station on that interface. If you want long and/or frequent transmissions, you can set up a repetitive ping with specified length and repetition rate:

ping <hostid> <length> <interval>

Note that the *length* parameter is the number of bytes of *data* included in the outgoing packet, and does not include the TXDELAY/TXTAIL intervals, AX.25 framing, or IP header. You can set it to "0" and get a minimum-length packet. Also note that the *interval* is in milliseconds, not seconds! It is okay (and maybe preferable) to ping a nonexistent host, but be aware that the pings will not go out unless there are entries in your routing and ARP tables that apply to that host. Let's suppose that you want to transmit pings that will give you an approximately two seconds on, two seconds off duty cycle for test purposes (don't keep this up

for long on a busy channel!). You could set the TXDELAY to give the desired transmission length, but you might forget to reset it later! We want to transmit about 2400 bits to get the two second transmission length. With the default settings, the TXDELAY interval accounts for about 400 bit times. The framing and headers account for about 300 more, so we need 1700 bits, or about 200 bytes of data. Assuming that the interface is called "pi0b", let's pick a test IP address and set up for the ping:

route add 44.128.1.1 pi0b
arp add 44.128.1.1 ax25 test-0 pi0b

Now you're ready to start the ping:

ping 44.128.1.1 200 4000

When you're done, hit F10 and enter "close" at the net> prompt.

If you are using a radio that has a combined microphone/PTT input and the keying is not working, you may need to change the value of R28. However, the value provided should work for nearly all applications. If the keying is erratic or the radio seems to stick in transmit mode, you probably have RF getting into the key line. Make sure that the antenna is well separated from your radio/computer setup (i.e., don't use a rubber duckie!). Persistent problems can usually be cured by properly shielding the cable between the PI2 card and the radio, and by adding bypass capacitors and RF chokes at the ends of the cable. The snap-together toroid chokes available at Radio Shack and other sources are a convenient means of reducing the RF pickup on cables.

Assuming that keying is okay, you can can begin increasing the transmit audio level by turning R26 clockwise. You can check the level with a scope on TP2. At this point you should monitor your transmitted signal with a second receiver. Although it's best to check the deviation of your signal with a deviation meter, you can generally get close to the right setting by increasing the transmit audio level until you are hearing no further increase in the audio level in the monitor receiver, and then backing it off until you start to hear a drop in the level. If the signal sounds clean and free of hum and distortion, then you're done.

Receive Adjustments

Receive Audio Level

The modem's receive audio input is low-impedance and will normally be connected to the speaker output of the radio. The level at the modem input can be measured at test point TP1. This level should be set to about 0.3 V peak-to-peak (if you're using a voltmeter that gives average or RMS readings, set it to about 0.1 V). The level should not exceed 0.75 Vp-p. If you are using a speaker output, try to arrange that the volume control setting cannot be easily changed once you've set it.

RX Bias Level

The RX bias adjustment sets the slicer level in the modem demodulator. The object of the adjustment is to set the bias so that the demodulated "0" and "1" bits have the same duration. The bias voltage is set by trimpot R45 (the one nearest the mounting bracket of the card) and can be measured at the RXB test point, TP3. The bias can be coarsely adjusted by setting it to 2.75 volts. The modem should work okay with this setting, but for optimal performance, it is recommended that you make a fine adjustment using one of the

two procedures outlined below. Both methods require that you have access to the receive data (RXDB) output of the modem. This signal is available on test point TP4.

Method 1: Generate a 1200 bps AFSK signal modulated by an alternating bit sequence (01010...) and input it to the modem at the level which you set up previously. One way to do this is to use a TNC with standard AX.25 firmware installed, either hooked up locally or received over the air from another packet station. Use the 'calibrate' command and the 'D" subcommand to generate the desired signal. Then, while observing the RXDB signal with an oscilloscope (it should have a rectangular waveform and a 600 Hz repetition rate), adjust the RX bias pot R45 until the signal has a 50% duty cycle. If you cannot easily generate the 01010... pattern, you can try making the bias adjustment while receiving random packet signals - it's a bit more difficult, but possible. Identify the "single bit" elements in the received data stream, and then adjust the bias until the '0' and '1' bits are the same duration (0.833 ms).

Method 2: If you can generate a sine wave with a frequency that is known reasonably precisely, you can adjust the RX bias by injecting a 1700 Hz sine wave (near the nominal 0.3 Vp-p level) into the modem audio input. The RXDB signal will either be a logic 'low' (near zero volts) or a logic 'high' (more than 3 volts). Adjust the RX bias pot R45 until the level just flips to the other state.

Modem Carrier Detect Indicator

The carrier detection (DCD) circuit requires no adjustment. However, it is very useful to have a visible indication of DCD activity, and a LED driver (Q2) has been provided for this purpose. The LED should be connected to J1, with the anode of the LED connected to the pin nearest the back (bracket end) of the board. You may already have a LED indicator on your PC which can be used for this purpose - for example, most PCs have both a 'turbo' switch and a 'turbo' LED. The LED is redundant if the 'turbo' setting is evident from the switch position. The polarity of the LED may not be evident, but you can just try it both ways on J1 and find the one that works. If you want to mount the DCD indicator externally, you could use some spare pins on the DB25 connector to provide the hookup.

Appendix C - Interfacing External Modems

GRAPES (WA4DSY) 56 kbps Modem

This modem provides both receive and transmit clocks, so the PI2 card is used in external clocking mode. The interface is TTL, but if your modem must be located a significant distance (i.e, more than a meter or two) from your PC, you should seriously consider installing RS422 line drivers and receivers and running a balanced interface. If your modem uses the 9-pin "D" connector as shown in the GRAPES documentation, then the wiring of the interface cable should be as follows:

Signal	P1 of PI2 Card	GRAPES 9-pin Connector
RXC	2	3
RXD	3	2
TXC	4	5
DCD	5	8
TXD	6	4
RTS	7	1
GND	9	6

The RTSAA jumper on the PI2 card should be in the open-collector position (upper pair of pins). If you choose to use the PTT output (pin 1) of the PI2 to key the transmit converter, you may wish to bring this line out to the modem, and then loop it through to the converter in place of the modem's PTT line. This allows the oscillators on the transmit side of the modem to run continuously, which substantially reduces the required keyup time; however, you do have to be careful to avoid interference from these oscillators into the receive side of the modem. This is easier, of course, if the receive and transmit frequencies of the modem are different, as would usually be the case if you have a full-duplex repeater. The other thing to watch for is interference into the 10.7 MHz IF of the modem's demodulator from the third harmonic of the oscillator in the baud rate generator (on the encoder board).

Kantronics D4-10

The Kantronics D4-10 440 MHz radio is frequently used as an RF modem, usually at 19.2 kbps. It does not do receive clock (bit timing) recovery, nor does it generate a transmit clock, so these functions must be supplied by the PI2 card, by running it in internal clocking mode. This means the baud rate of the A port should be set to *19200* in the *attach pi* line. The cabling between the PI2 and the D4-10 is as follows:

Signal	P1 of PI2 Card	D4-10 TTL Port
PTT	1	3
RXD	3	5
DCD	5	2
TXD	6	1
GND	9	6,9

The RTS line (with RTSAA jumper set for open collector) can be used instead of the PTT. Thanks to John Ackermann, AG9V, for supplying this information. John also provides these tips:

A TXDelay of 10ms will work for a pair of D4-10s; 20-30ms is probably safer. If you're talking to other radios that don't have 1ms timer granularity (like DataEngines running BPQ), 40 or 50ms works well for a network-wide TXDelay setting.

TXTail needs to be set to 3ms for reliable operation. 1ms may seem to work, but will generate excessive bad packets.

[The following refers to the original PI card. The interrupt load caused by noise on the RXD line is not a problem with the PI2, since RXD is gated by DCD on the PI2. However, the adjustments described should be of interest to all D4-10 users.]

The garbage on the RXD line created by noise when there's no signal present can result in the PI card thinking it sees huge numbers of bad packets, and thus generating huge numbers of spurious interrupts. These can bog the system down dramatically, even when NOS isn't running -- we've seen the screen lag keyboard input by better than 500ms. This is primarily caused by the D4-10 data slicer threshhold control, R17, being misadjusted. To adjust it, hook an oscilloscope to the RXD line, set it for 1V/cm and a normal audio sweep rate, and look at the noise when there's no signal present. You should see two "rails" at 0 and 5 volts, with noise between them. Adjust R17 so the intensity of both rails is the same. This puts the data slicer threshhold in the middle of the noise bandwidth, and in addition to reducing the spurious interrupts, also optimizes receive performance.

Earlier vintage D4-10s had a wiring error that caused adjustment of the data slicer threshhold to shift the receiver frequency. This is cured in later production (if your radio has one jumper and one small cap on the bottom of board, you're all set. If you don't have the Kantronics mods, you should re- adjust the receiver crystal after you adjust the threshhold. Inject an on- frequency signal, modulated with 3kHz of 1000 hz audio tone. Look at the RXD line on the scope, You should see a square wave. Adjust the appropriate coil (RL1 or RL2) until the square wave is 50% duty cycle (ie, the top and the bottom are the same width). You've now put the RX on frequency.

You can also clamp the RXD line so there's no jitter on it during no-signal conditions. That way you'll be <sure> not to overrun the PI card (though this isn't an excuse for a misadjusted threshhold). It's easy to clamp the RXD line with DCD. Just put a diode (1N914 or similar) between TTL port pin 2 and U1 pin 2. The cathode end should point at U1 pin 2. You can tack the diode between the appropriate pins on the bottom of the D4-10 circuit board. [Again, this modification is not needed with the newer PI2 card.]

TAPR 9600 bps Modem

Of the various 9600 bps modems available, the TAPR modem kit is the least expensive, and also the most versatile. Interfacing it to the PI2 is straightforward; it requires preparation of a cable to connect the **PI2 A port** to the modem's disconnect header (P3). The connections are as follows:

Signal	P1 of PI2 Card	Modem Disconnect (P3)
RXD	3	17
DCD	5	1
TXD	6	19
RTS	7	5
32CK*	8	11 [Not needed if the modem has the "clock option]
GND	9	15

The PI2 card is operated in internal clocking mode. The details of the setup depend on whether the onboard clock option is installed on the modem:

No clock option installed. In this case, the PI2 must supply a 16X clock to the modem. These steps are necessary:

(1) Remove the TXCK jumper on the PI2 card.

(2) Connect P1-8 of the PI2 to P3-11 on the modem, as shown in the table above.

(3) Install a jumper at JP3 on the modem card, to select a 16X clock. No other jumpers need to be installed.

(4) Specify the 'g' option when you set the speed parameter for the A port in the 'attach pi' statement, e.g.:

attach pi 0x380 7 1 ax25 9k6$1k2 552 552 9600g 1200

Clock option installed. In this case, the modem supplies its own 32X clock, so none of the steps outlined above are needed. Leave the PI2 TXCK jumper in place, do not make any connection to P3-11 on the modem board, and do not include the 'g' option in the 'attach pi' line. On the modem board, install a jumper at JP2 to select internal clocking.

You can also try interfacing a 9600 bps modem to the **PI2 B port** if the internal modem is not installed. Since this port is interrupt-driven, performance is highly dependent on the speed of the host machine, as well as many other factors (the version of software driving the card, what other interfaces are attached, etc.). Don't bother trying it on an XT! The connections are as follows:

Signal	P2 of PI2 Card	Modem Disconnect (P3)	
RXD	3	17	
DCD	8	1	
TXD	2	19	
RTS	4	5	
TXC*	24	11	[Not needed if the modem has the "clock option"]
GND	9	15	

Other 9600 bps Modems

We have not had the opportunity to check other 9600 bps modems, but there should not be any problems if they are based on the G3RUH design. Use the connections for the TAPR modem above as a guideline.

Appendix D - Installing the Optional RS422 Interface

To install the RS422 interface, do the following:

(1) Install a 16-pin DIP socket at U14, and install a 26LS31 line driver chip in the socket.
(2) Install a 16-pin DIP socket at U17, and install a 26LS33 (or 26LS32) line receiver chip in the socket.
(3) Install 120 ohm resistors at R41, R42, R43, and R44.
(4) Remove the TTLRS jumper (J2).
(5) Make sure that the RTSAA jumper (J4) is in the TTL position (lower two pins).

To return the port to TTL operation, simply reinstall the jumper at J2.

Please note that the DCD input and RTS output signals are active low, so that the sense of the corresponding RS422 signals is reversed from the other signals.

Appendix E - Hints and Kinks

Troubleshooting

Problems making the PI2 board work are often the result of hardware or software conflicts in your PC. Before installing the board, it is worthwhile to take a few minutes to examine your system carefully, making note of any usage of IRQs, I/O addresses, and DMA channels. The existence of another board which can use the same IRQ as the PI2 can be a problem, even if that board is not actively using it. Make sure the other board's connection to that IRQ line is really broken!

We have encountered several PC motherboards whose bus timing causes problems for the PI2's DMA interface. Many of the systems with problems have had 386SX processors, but beyond that there is no apparent pattern. The most common symptom is a failure to transmit decodable packets, whereas receiving packets works okay. A fix which oftens works for this problem is to increase the delay of the DDIOW signal. To increase the delay, move jumper J13 to a position farther to the right (i.e., from Q1 to Q2, and then to Q3 if the problem persists).

Conflicts with other software can also produce frustrating problems. If your PI2 interface does not seem to be working properly, remove all of your TSR programs, and all of the device drivers loaded in your CONFIG.SYS file. Then try running NOS and see if the problem has cleared up; if it has, put the other software back in, one by one, until you determine which one is causing the problem. One likely source of problems is memory managers. Using Microsoft's HIMEM.SYS to load the DOS kernel high seems to work fine, but the more complex memory managers such as Quarterdeck's QEMM can be more problematic. If you use the PI2 DMA port (A port), then you **must** specify the LOCKDMA parameter in your QEMM entry in CONFIG.SYS. Even with that parameter, performance of the DMA port with QEMM loaded has proved to be erratic, but your mileage may vary... we welcome your observations! Similarly, we would appreciate reports of performance of the PI2 when NOS is run in multitasking DOS environments such as Windows and OS/2. We know that the PI2 DMA interface does not work in an OS/2 VDM using the DOS PI driver (however, there is now a PI driver for OS/2 which works very well).

Finally, make sure you use good **grounding** practices. Poor grounding can cause frustrating noise problems whose symptoms often masquerade as other problems. Use braid to connect grounded points on

your PC, external modems, and RF hardware, and, if possible, also tie the braid at one point to a good external ground.

Back-to-Back Connections

It is sometimes useful to connect a pair of PI2 boards back-to-back in order to check that they are functioning properly. The steps required to do this are as follows:

(1) Construct a 'null modem' cable which provides a ground connection plus crossovers which connect the RXD line on each board to the TXD line on the other, and the RTS line on each board to the DCD line on the other.

(2) Configure the RTS jumper on each board for TTL output.

(3) Attach the PI2 interface at each end, specifying internal clocking (i.e., specify a nonzero baud rate for the port used in the back-to-back connection). A typical NOS attach statement might be

attach pi 0x380 7 1 ax25 pi0 2088 2088 19200 1200

for a 19.2 kbps back-to-back connection between the high-speed 'A' ports. Note that in internal clocking mode, the maximum baud rate is 57.6 kbps. The interrupt-driven 'B' port will probably not work well at the higher rates.

The above instructions assume the use of the built-in PI driver for NOS. As noted in the section of this documentation which deals with the PI packet driver, the latter driver offers an additional clock mode which makes higher-speed back-to-back hookups possible (for the high-speed 'A' ports *only*). In this mode, each board provides an external receive clock signal for the other board. If you try this mode, in addition to loading the packet driver and providing the appropriate attach statement in NOS, you must take two additional steps:

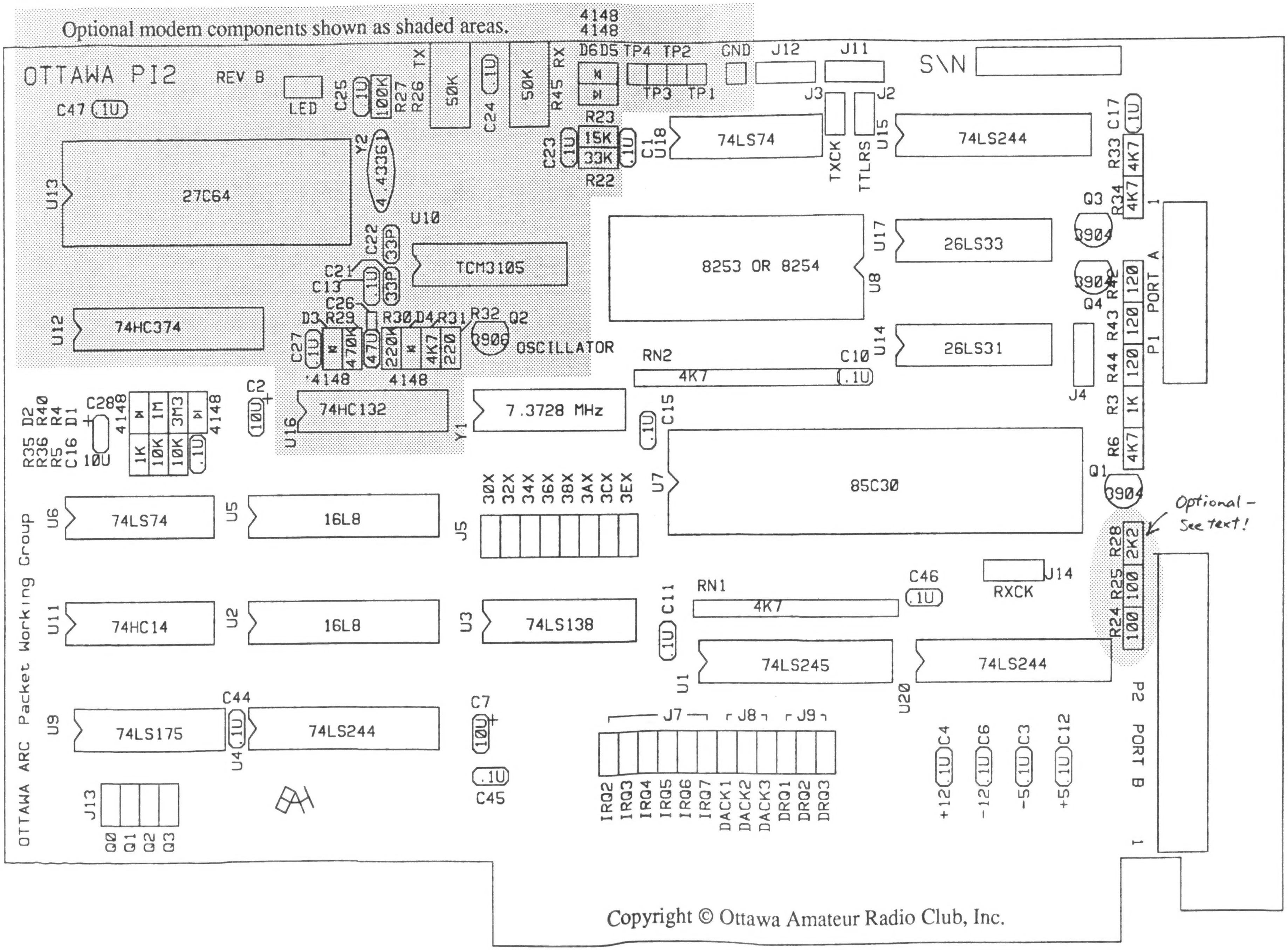
(1) Remove the TXCK jumper J3 (don't forget to reinstall it later, if you'll be using the board with a modem that provides a transmit clock signal, such as the GRAPES modem).

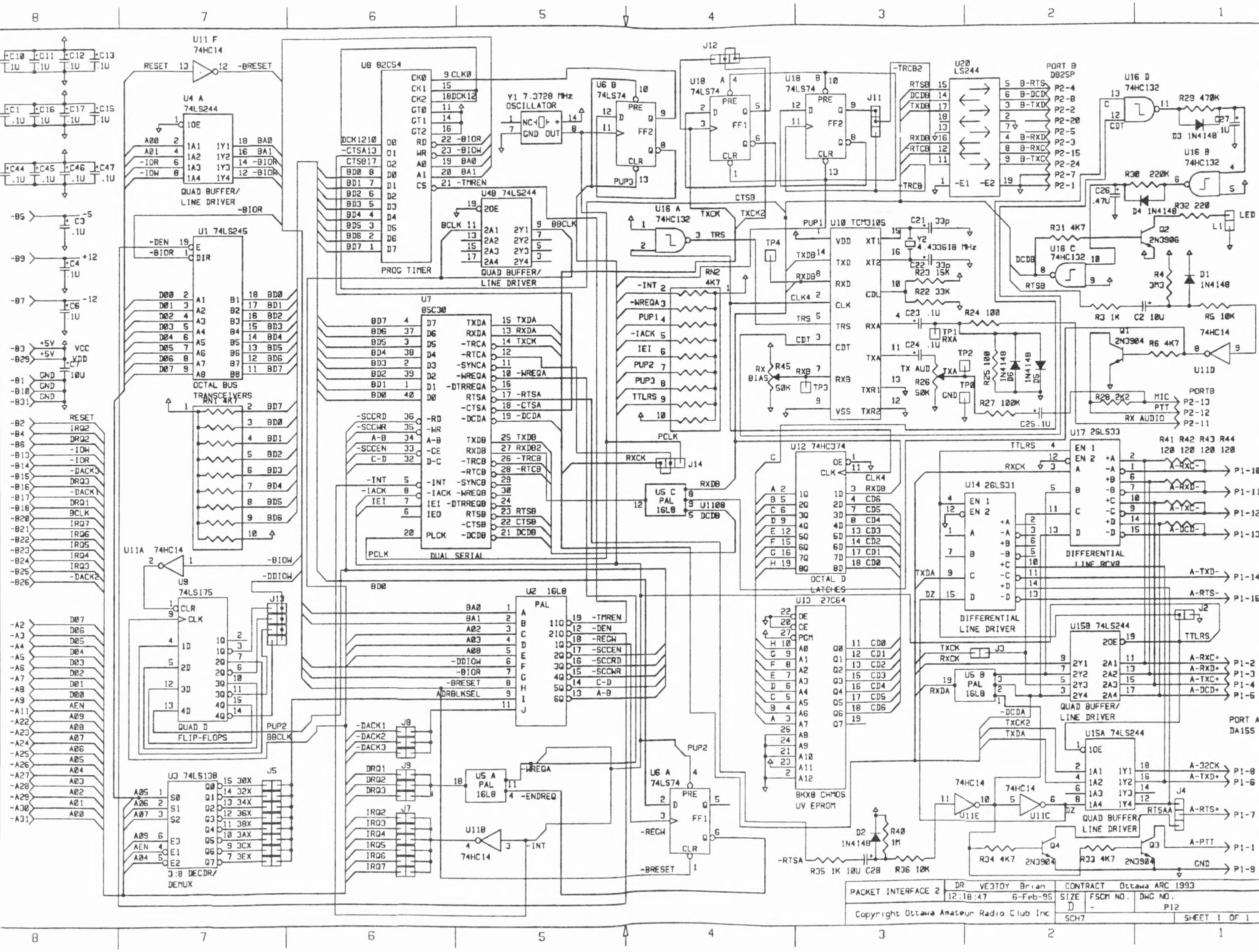
(2) Add to your null modem cable, crossovers between the RXC and 32CK signals. The complete cable then has the following connections:

2 --- 8 (Needed only for the special packet driver clock mode described above)
3 --- 6
5 --- 7
6 --- 3
7 --- 5
8 --- 2 (Needed only for the special packet driver clock mode described above)
9 --- 9

Appendix F - Schematic and Parts Placement Drawing

The PI2 schematic and the placement of parts on the board are shown on the following two pages.


Optional modem components shown as shaded areas.
OTTAWA PI2
REV B
OTTAWA ARC Packet Working Group
Copyright © Ottawa Amateur Radio Club, Inc.
27C64
74HC374
TCM3105
8253 OR 8254
74LS74
74LS244
26LS33
26LS31
74HC132
7.3728 MHz
85C30
74LS74
16L8
74HC14
16L8
74LS175
74LS244
74LS138
74LS245
74LS244
OSCILLATOR
S\N
PORT A
PORT B
LED
TXCK
TTLRS
RXCK
GND
IRQ2
IRQ3
IRQ4
IRQ5
IRQ6
IRQ7
DACK1
DACK2
DACK3
DRQ1
DRQ2
DRQ3
30X
32X
34X
36X
38X
3AX
3CX
3EX
+12
-12
-5
+5
Optional - See text!


PACKET INTERFACE 2
DR VE3TOY Brian
CONTRACT Ottawa ARC 1993
12:18:47 6-Feb-95
SIZE D FSCM NO. DWG NO. P12
SCH7
SHEET 1 OF 1
Copyright Ottawa Amateur Radio Club Inc
U7 85C30 DUAL SERIAL
U8 82C54 PROG TIMER
U10 TCM3105
U13 27C64 8Kx8 CHMOS UV EPROM
U12 74HC374 OCTAL D LATCHES
U14 26LS31 DIFFERENTIAL LINE DRIVER
U17 26LS33 DIFFERENTIAL LINE RCVR
U1 74LS245 OCTAL BUS TRANSCEIVERS
U3 74LS138 3:8 DECDR/DEMUX
U9 74LS175 QUAD D FLIP-FLOPS
U2 16L8 PAL
Y1 7.3728 MHz OSCILLATOR
Y2 4.433618 MHz
PORT A DA15S
PORT B DB25P